Robot learning from demonstrations:

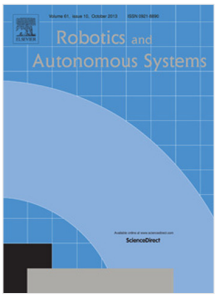
Ghalamzan Esfahani, Amir Masoud; Ragaglia, Matteo

DOI:
10.1016/j.robot.2017.12.001

Document Version
Peer reviewed version

Citation for published version (Harvard):
Ghalamzan Esfahani, AM & Ragaglia, M 2018, 'Robot learning from demonstrations: Emulation learning in environments with moving obstacles', *Robotics and Autonomous Systems*, vol. 101, pp. 45–56. https://doi.org/10.1016/j.robot.2017.12.001

Accepted Manuscript



Robot learning from demonstrations: Emulation learning in environments with moving obstacles

Amir M. Ghalamzan E., Matteo Ragaglia

PII: S0921-8890(17)30298-1
DOI: https://doi.org/10.1016/j.robot.2017.12.001
Reference: ROBOT 2959

To appear in: *Robotics and Autonomous Systems*

Received date : 14 May 2017
Revised date : 28 October 2017
Accepted date : 4 December 2017

Please cite this article as: A.M. Ghalamzan E, M. Ragaglia, Robot learning from demonstrations: Emulation learning in environments with moving obstacles, *Robotics and Autonomous Systems* (2017), https://doi.org/10.1016/j.robot.2017.12.001

Robot Learning from Demonstrations: Emulation Learning in Environments with Moving Obstacles

Amir M. Ghalamzan E.[a], Matteo Ragaglia[b]

[a] *University of Birmingham, Edgbaston, B15 2TT, Birmingham, United Kingdom*
[b] *Yanmar R&D Europe, Viale Galileo 3/A, Firenze, 50125, Italy*

Abstract

In this paper, we present an approach to the problem of Robot Learning from Demonstration (RLfD) in a dynamic environment, i.e. an environment whose state changes throughout the course of performing a task. RLfD mostly has been successfully exploited only in non-varying environments to reduce the programming time and cost, e.g. fixed manufacturing workspaces. Non-conventional production lines necessitate Human-Robot Collaboration (HRC) implying robots and humans must work in shared workspaces. In such conditions, the robot needs to avoid colliding with the objects that are moved by humans in the workspace. Therefore, not only is the robot: (i) required to learn a task model from demonstrations; but also, (ii) must learn a control policy to avoid a stationary obstacle. Furthermore, (iii) it needs to build a control policy from demonstration to avoid moving obstacles. Here, we present an incremental approach to RLfD addressing all these three problems. We demonstrate the effectiveness of the proposed RLfD approach, by a series of pick-and-place experiments by an ABB YuMi robot. The experimental results show that a person can work in a workspace shared with a robot where the robot successfully avoids colliding with him.

Keywords: Robot Learning from Demonstration, Dynamic Environment, Moving Obstacles

1. Introduction

State-of-the-art Robot Learning from Demonstration (RLfD) approaches (e.g. Gams et al., 2015; Rai et al., 2016) enables a robot to learn a task and obstacle avoidance model from human demonstrations. Nevertheless, these approaches do not tackle the problem in varying environments where humans move objects while the robot is performing the task. As the common vision for the future of robotics suggest robots should work in non-stationary environments in collaboration with humans (e.g. Maeda et al., 2017), we present an incremental approach to RLfD (Incremental Robot Learning from Demonstration (IRLfD)) that enables a robot to learn from demonstrations: (i) a task model; and (ii) a control policy for avoiding stationary obstacles. In addition, we extend the control policy such that the robot is capable of avoiding moving obstacles.

For instance, if we consider a human provides a YuMi robot with sufficient demonstrations to perform a pick-and-place task (as shown in Fig. 1), this exemplifies a pos-

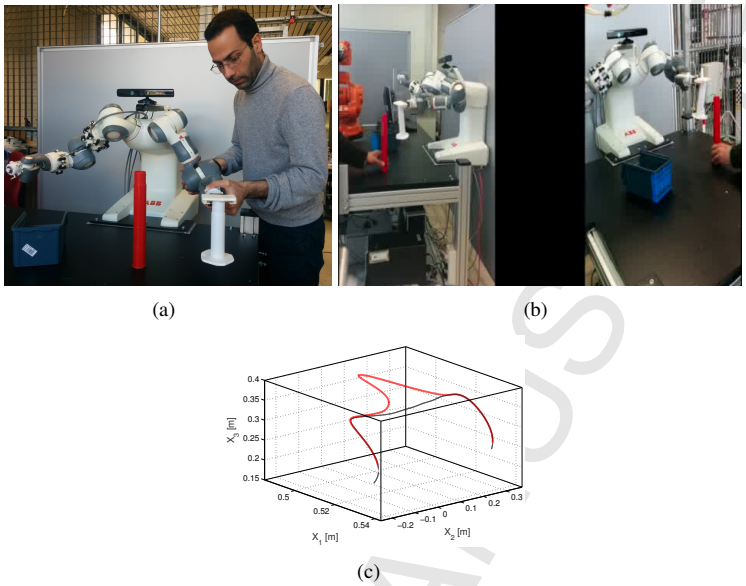


Figure 1: (a) a human demonstrates the manipulation task by the ABB YuMi robot. A cylindrical object is considered as an obstacle fixed on the table during the demonstration; (b) the robot performs the task while avoiding a collision with the moving obstacle using the model learnt from the demonstration. Although the obstacle is intentionally moved by human to make collision with the robot, the proposed approach effectively deals with it and allows a successful completion of the task without any collision. (c) shows the trajectory of the pick-and-place task with an stationary obstacle (red line) and without an obstacle (black line).

sible situation in a future production line in which a human needs a robot to take over his job (e.g. picking up an object, which moves on a conveyor belt, and placing it in a box) for a few hours. Thus, to teach the task to the robot he demonstrates the task a few times (Fig. 1(a)). Then, the robot is desired to adaptively perform the task in collaboration with other human workers; i.e. robot performs the task while humans are changing the state of the environment by moving objects in robot workspace. Teaching a robot how to perform the pick-and-place task in a dynamic environment is very challenging and demanding (Wheeler et al., 2002). Here, we assume that a sophisticated vision system in real time provides the robot with the state of the environment, namely start and goal points and the position of stationary and moving objects.

The contribution of this paper is thus the incremental robot learning from demonstration for performing the demonstrated task in a dynamic environment (IRLfD-DE). In IRLfD, a robot learns (i) a task model and (ii) a control policy only to avoid colliding with stationary obstacles. The IRLfD-DE presented in this paper generalises the model such that it can generate a trajectory needed to perform the task in an environment with moving obstacles. By contrast, other approaches (e.g. Gams et al., 2015; Rai et al., 2016) including IRLfD enable a robot to perform the task just in the presence of stationary obstacles using the model learnt from demonstrations.

We extend the IRLfD by (i) using an object tracker from OpenCV, which provides

us with a reasonably fast real time estimation of an obstacle's position; (ii) utilising a linear Kalman filter providing a prediction of future positions of the obstacle in a time horizon and filtering the noise from the obstacle's position estimated by the object tracker; and (iii) by reformulating the utility function of the IRLfD to deal with the uncertainty of the estimated obstacle's position. The obtained control system robustly copes with moving obstacle while performing a task by the model learnt from demonstrations. We demonstrate the effectiveness of the extended approach by the pick-and-place example, shown in Fig. 1(a); where the robot successfully performs the task using the learnt model while a human is moving an object in the workspace of the robot. The movements of the obstacle may not be linear, as shown in the accompanied video of the experiment with the YuMi robot. Although we considered a linear model in the control system, our experiment shows the robot successfully performed the task while a human was moving an object in the robot's workspace, due to the uncertainty included in the learnt model.

The remainder of this paper is organised as follows: firstly, we present an overview of the works on RLfD in section 2 and the relation between the observational learning and the IRLfD-DE in section 3; then, the IRLfD is formulated in section 4 and it is extended to a dynamic environment (IRLfD-DE) in section 5. The effectiveness of IRLfD-DE is demonstrated by series of experiment in section 6. In section 7, we discuss the performance of the approach and the experimental results presented in section 6. Finally, we present our conclusions and propose some directions for future works in section 8.

2. Related works

The problem of generating a trajectory necessary and sufficient for a robot to perform a task while it avoids having a collision with obstacles has been studied in the domains of the relevant literature: e.g. (i) motion planning, such as RRT* (Karaman and Frazzoli, 2010), CHOMP (Ratliff et al., 2009) and STOMP (Kalakrishnan et al., 2011); and(ii) robot learning from demonstration (Rai et al., 2014; Byravan et al., 2015; Gams et al., 2015; Tanwani and Calinon, 2016). The optimisation based approach has been illustrated to be effective in many robotic contexts, including industrial manipulation where the manipulator's workspace condition is fixed and known a priori. However, it may not be pragmatically useful where a robot needs to perform different tasks as per user need and desire, in an unknown environment and in collaboration with humans. In this regard, RLfD (e.g. Kober et al., 2012; Krug and Dimitrov, 2015) has been shown to be practically a useful means of providing a user with the flexibility of programming a robot by only demonstrating the desired performance behaviour of a new task.

For instance, Tanwani and Calinon (2016) studied the semi-tied Gaussian mixture models for learning a robotic manipulation task from multiple demonstrations, where the robot can adaptively replicate the task in a different situation, including different start and goal poses and with a simple stationary obstacle. Furthermore, *inverse optimal control* (IOC), also known as inverse reinforcement learning, learns a utility function from demonstrations. This utility function is then used to generate actions every time a robot must perform the demonstrated task (Ng and Russell, 2000;

Abbeel and Ng, 2004; Ziebart et al., 2008). However, the computational cost is a major challenge in IOC. Byravan et al. (2015), for example, proposed an approach of IOC which was utilised for manipulating an object in a cluttered and stationary environment. Although IOC approaches (e.g. Tanwani and Calinon, 2016; Byravan et al., 2015) showed promising results for learning a model of a task and obstacle avoidance from demonstrations, they only showed successful experiments with stationary constraints/obstacles.

Dynamic movement primitives (DMP), also known as imitation learning, enable a robot to learn skills very similar to those of humans. These DMP aim at producing a trajectory in *real-time* similar to that which is demonstrated. There are a number of methods that combine DMP with a defined control policy of avoiding a collision with an obstacle (e.g. Calinon et al., 2010; Guenter et al., 2007; Kormushev et al., 2010; Park et al., 2008). For instance, Hoffmann et al. (2009) added an acceleration term to the equations of motion in the DMP formulation to avoid a collision with a moving obstacle. Although these proposed approaches enable a robot to avoid colliding with an obstacle with a predefined obstacle avoidance control policy, a non-expert cannot teach a robot his/her own desired policy of avoiding a collision with different objects just by demonstration.

A stream of research learns the parameters of a coupling term of the obstacle avoidance control policy added to the DMP formulation. For example, Fajen and Warren (2003) proposed a 2-D dynamical model to describe human behaviours of steering towards a goal point while avoiding a collision with an obstacle. This obstacle avoidance function is used as a repulsive force term in combination with DMP (Pastor et al., 2009). Moreover, Rai et al. (2014) extended this result to 3-D and augmented the reactive feedback coupling term with DMP, where the corresponding parameters are learnt from human demonstration. They show the DMP converge to the goal point if the obstacle is stationary. Gams et al. (2015) learned the coupling term of feedback and feedforward control from human demonstrations by a statistical learning approach. Rai et al. (2016) used DMP and neural networks to represent a skill and to learn a policy from human demonstrations, consecutively. However, these works propose a solution that results in a general modification of the demonstrated trajectory, despite the fact that in some cases a local deviation from an obstacle-free trajectory is necessary and sufficient for a robot to perform a task, e.g. the pick-and-place task with the YuMi robot.

By contrast, Ghalamzan et al. (2015) presented a modular approach, namely IRLfD, that results in local modification of the trajectory for avoiding an obstacle. This method includes: (i) regression, (ii) DMP and (iii) obstacle avoidance modules. Although one could use a unified approach combining all the modules, generalisability is the main benefit of keeping independent modules of RLfD. Furthermore, the precision and performance of every module can be independently analysed. On the contrary, an end-to-end solution learning a model for a specific problem, e.g. Levine et al. (2016); Gu et al. (2016), may not be easily analysed and/or its parts cannot be used for different tasks.

Although these existing methods learn efficiently the models of the task and obstacle avoidance from demonstrations, they do not face a moving obstacle. Here, we extend the IRLfD such that our novel approach copes with moving obstacles in addition

to stationary obstacles. Thus, it learns a model of the task and an obstacle avoidance policy from human demonstrations and generalises it to a scene[1] with a moving obstacle. Here, we only discuss the problem of collision avoidance in robot's operational space between a moving obstacle and an object to be manipulated during the task execution. Obstacle avoidance in the configuration space can be successfully dealt with using a motion planner, (e.g. Ratliff et al., 2009; Kalakrishnan et al., 2011). In fact, the obstacle avoidance behaviour in the configuration space cannot be human-like because the robot's and the human's body are not identical. Therefore, obstacle avoidance in the configuration space is always optimally programmed beforehand, e.g. by using the kinematic redundancy of the robot. On the contrary, to avoid a collision between an obstacle and the object to be manipulated is most often task specific and a user usually needs to program it very fast in the robotic site. Our approach provides a robot with this capability and allows a user to teach the robot a task and obstacle avoidance behaviour by demonstration; whilst the robot can generalise the learnt model to new start and goal points in an unseen environment with stationary and moving obstacles. Generalisation of the learnt model to an environment with a moving obstacle is the main contribution of this paper. We show the effectiveness/usefulness of the proposed approach by a series of manipulation experiments in an environment with a moving obstacle.

3. Relation between IRLfD-DE and observational learning

Schaal (1999); Billard et al. (2008); Argall et al. (2009) present broad overviews of research on generalising demonstrations to a new environment. Thanks to these studies and according to the studies of human Observational Learning (OL) (e.g. Thompson and Russell, 2004; Whiten et al., 2009), we identify different levels of generalising a demonstrated task under different conditions of an environment: (i) *mimicking*, (ii) *imitation* and (iii) *emulation*. This indicates that humans (i.e. learners) do not learn a single complex model from demonstrations (i.e. a behaviour model). Similarly, we consider three modules of robot learning from demonstrations (Fig. 2). Mimicking (Whiten et al., 2009) is the copying of a model's body movements. Thompson and Russell (2004) identified that mimicking must involve no conceptualisation by the observer concerning the purpose of the action. Hence, it may not be possible to accomplish the task in a new environment through the mimicking alone. In the early 60s, robots used a recorded trajectory to perform a task in a fixed industrial environment (Tanner, 1981). This corresponds with the *mimicking* in OL. For example, a new position of the object to be picked causes a failure when repeating the pick-and-place by using a recorded trajectory. A regression approach (e.g. Calinon, 2009; Billard et al., 2016) can be used to compute a smooth nonlinear principal trajectory from a dataset collected from noisy human demonstrations. We recognise such approaches as a mimicking RLfD.

Imitation learning has been defined by Whiten and Ham (1992) as a goal-oriented copying the form of an observed action. In imitation learning, an observer is assumed to be able to recognise what the form of the model's movements is bringing about and use that to carry out the task. This is analogous to a traditional dynamic movement

[1] The terms 'scene' and 'environment' are interchangeably used throughout this paper.

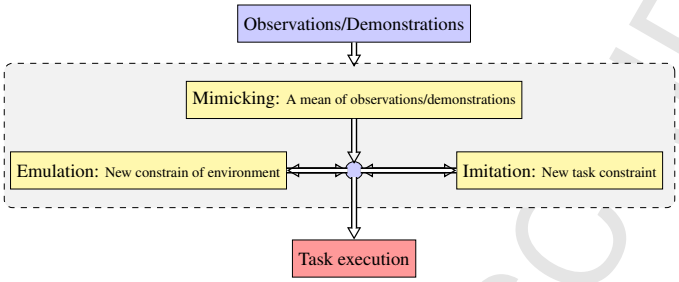


Figure 2: A schematic of the learning processes recognised in observational learning such as mimicking, imitation, and emulation. The arrows show information follows; whereas double-sided arrows show that information flows in both directions. In the mimicking, an agent just copies a mean of observations without preserving the purpose of the action. Hence, it may not successfully perform a task if the environment changes. In the imitation, the learner recognises what the form of the model's movements is bringing about and uses that to carry out the task. In the emulation, the observer replicates the expected results of the model's action. The execution is a result of combining all these components.

primitives (Schaal, 1999) which can generalise a trajectory such that it satisfies new task constraints, e.g. a new goal point. It has been proved in neuroscience (e.g. Thoroughman and Shadmehr, 2000) that the brain can produce desired motor commands by combining motor primitives. This is the core idea of DMP.

In *emulation learning*, the observer replicates the expected results of the model's action (Whiten et al., 2009). In the motivating example presented in the introduction section, emulation corresponds with learning the capability of adapting the trajectory to suit an environment with some obstacles from the demonstration.

Lastly, many intelligent and context-specific responses of humans to different stimuli during a task execution are consistent with the theoretical framework of optimal control (Todorov, 2004) based on the studies of sensorimotor learning in cognitive psychology and neuroscience. Prediction provides the brain with expected sensory consequences. The corresponding motor commands are then used to obtain the desired consequences (Flanagan et al., 2003). Likewise, to perform the task (learnt from demonstrations) in a dynamic environment, we propose a system, including state predictions and the control system using these predictions for generating necessary actions for performing a task robustly.

4. Problem formulation

In the pick-and-place example (1(a)), the robot repeats the task by using a fixed sequence of end-effector poses in an invariant environment. Performing the task in this *nominal environment* imposes some constraints on the robot movements, which is denoted by C_n and called *task constraints*. For instance, the initial and desired pose of the object determines the initial and terminal points of the trajectory. C_n is invariant across different demonstrations if the initial and desired location of the object to be manipulated does not change. In mimicking module of IRLfD, a recorded trajectory (called demonstration) satisfying C_n is thus sufficient for the robot to perform the task (this is

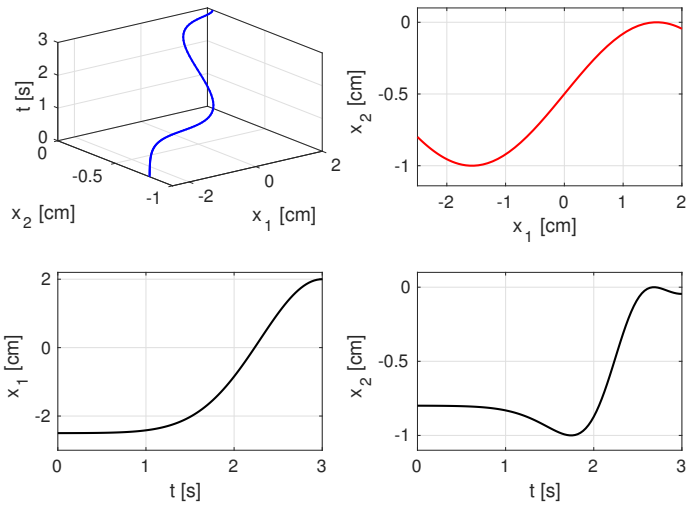


Figure 3: A sample trajectory of a robot's end-effector (top left) that is used as a nominal trajectory to perform a task; its corresponding 2-D path (top right); the trajectory along the first main axis (bottom left) and the trajectory along the second main axis (bottom right).

called mimicking). If the demonstration is noisy, a regression approach (e.g. Gaussian process) can compute a smooth trajectory from a few collected demonstrations.

Fig. 3 shows a simulated trajectory that is assumed to be sufficient for a robot to manipulate an object in a nominal environment. In the imitation module of IRLfD, a DMP model is trained by this smooth trajectory. The model is then used to generate a new smooth trajectory for a new start or terminal points (Fig. 10(e)). The task constraint C_n is determined by the initial and desired location of the object to be manipulated. Let's denote by ζ^N the smooth trajectory generated by either imitation or mimicking module and call it *nominal trajectory*, as shown in Fig. 3. The spatial part of the nominal trajectory, its tangent- and cotangent-space (normal to tangent space) is shown in Fig. 4(a).

We hypothesise that a Utility Function (UF) can generate a trajectory, denoted by ζ_u, that satisfies the constraints imposed by both a task and an environment. IRLfD showed that the parameters of a UF can be computed by minimising the distances between the demonstrated trajectories and the obtained trajectory. This utility function forms an attractive field around ζ^N as per eq. (1).

$$U_I(\mathrm{x}_k) = \left(\mathrm{x}_k - \mathrm{x}_k^{\mathrm{N}}\right)^T \left(\mathrm{x}_k - \mathrm{x}_k^{\mathrm{N}}\right) \tag{1}$$

where $\mathrm{x}_k^{\mathrm{N}} \in \zeta^N$, $1 \leq k \leq T_e$ and T_e is the number of samples of ζ^N. $\left(\mathrm{x}_k - \mathrm{x}_k^{\mathrm{N}}\right) \subset \mathrm{N}_k$ and N_k is orthogonal to a tangent space of ζ^N at x_k. The cotangent space of trajectory in Fig. 3 is shown with the red dash-dotted lines in Fig. 4(a). The quadratic utility function is an *imitation* module of IRLfD. It is denoted by U_I and for this example is visualised in Fig. 4(b). In addition, the solution to this UF (ζ_U), which is identical to ζ^N, is shown with the red line in this figure.

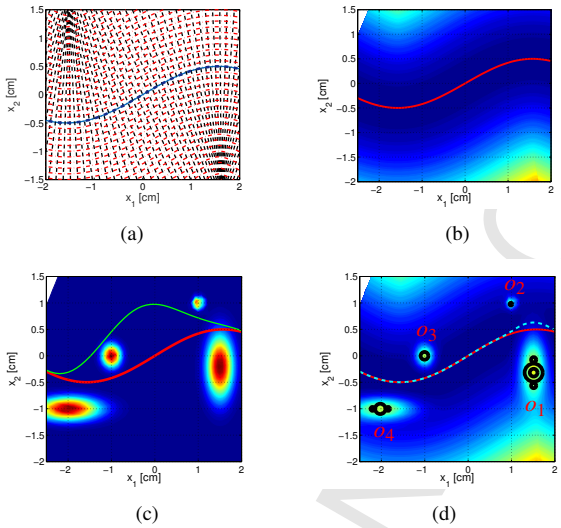


Figure 4: Nominal trajectory (blue thick line), its corresponding cotangent-space (black dashed line) and tangent-space (red dash-dotted line) are shown in Fig. (a). Fig. (b) shows a quadratic utility function formed around the nominal trajectory (red thick line) resembling a task constraint. The region with colder colour (dark blue) represents the utility with low-cost value. The red dash-dotted lines in Fig. (a) represent the level sets of this utility function. Fig. (c) represents the contour of the emulation utility function of four obstacles added to the environment. The emulation utility function represents the scene constraint of avoiding an obstacle in accordance with obstacle shapes, sizes and positions. In this figure, the nominal trajectory, that collides with the obstacle on the right-hand side of the figure, and an example trajectory, which avoids collision with the obstacles, are shown with red and green lines, consecutively. Fig. (d) shows the contour of the total utility function and a trajectory (red dashed line) that avoids a collision with the obstacle o_1, located on the right-hand side of the figure $\mathrm{y}_1 = [1.5, -0.25]$, while it has the minimum distances to the nominal trajectory.

By contrast, a set of constraints varying across different demonstrations is called *scene constraints*, denoted by $C_d, \forall\, d = 1, ..., n_{dem}$ where n_{dem} is the number of demonstrations. For instance, avoiding a collision with an obstacle poses constraints on robot movements during performing the pick-and-place example. In Fig. 4(c), different types of obstacles are added to the scene where their location may change across different task performances. In Fig. 4(d), this constraint results in a trajectory response in accordance with the obstacle position and orientation (y_1), and obstacle type, labelled by o_1. Nonetheless, the initial and terminal point of the trajectories in Fig. 4(b) and 4(d) are identical because these are the points at which the object to be manipulated must be picked up and placed. These constraints, namely task constraints, are independent of the positions of the obstacles, which are called scene constraints. Although ζ^N shown with the red line in Fig. 4(b) satisfies the task constraints, it does not satisfy the scene constraints caused by the obstacle shown by o_1 and results in a collision (Fig. 4(d)). A trajectory computed by a planner is shown by the green line in Fig. 4(c) that avoids the collision. However, this trajectory does not allow the robot to pick up and deliver the object at the desired poses, i.e. it does not satisfy the task constraints. A Gaussian term

is thus added to the UF which allows obstacle avoidance in IRLfD, as per eq. (2). This term is denoted by U_E and called emulation component of the UF and has a covariance matrix θ to be learnt from demonstrations.

$$U_E\left(\mathrm{f}_k : \zeta^N, \{\theta, \mathrm{y}\}\right) = e^{\left(-\mathrm{f}_k^T\ \theta^{-1}\ \mathrm{f}_k\right)} \tag{2}$$

where $\mathrm{f}_k = (\mathrm{x}_k - \mathrm{y})$ is a vector of the environmental features and y is the position of an obstacle. Parameter θ in eq. (2) allows for learning an influence of an obstacle on the UF in accordance with the shape and size of the obstacle. For each obstacle, one emulation term is added to the imitation utility function that is presented in eq. (2). Therefore, the final UF may include many emulation terms, according to the number of obstacles (n_{obs}) that exist in the environment, as per eq. (3). For instance, Fig. 4(d) visualises the UF for the task, whose nominal trajectory ζ^N is shown in Fig. 4(a), where 4 types of obstacles added to the scene.

$$U\left(\mathrm{x}_k : \zeta^N, \{\theta_j, \mathrm{y}_j\}_{j\in[1,n_{obs}]}\right) = \left(\mathrm{x}_k - \mathrm{x}_k^{\mathrm{N}}\right)^T \left(\mathrm{x}_k - \mathrm{x}_k^{\mathrm{N}}\right) + \sum_{j=1}^{n_{obs}} e^{\left(-{}_j\mathrm{f}_k^T\ \theta_j^{-1}\ {}_j\mathrm{f}_k\right)} \tag{3}$$

A solution trajectory to this utility function $\zeta_{U(\theta,\mathrm{y})}$ is the desired trajectory of the robot end-effector that satisfies task and scene constraints. We use ζ_U instead of $\zeta_{U(\theta,\mathrm{y})}$ for the sake of simplicity of the notation.

4.1. Computing Parameters of UF from Demonstrations

To perform the task without a collision with an obstalce in the robot's workspace, the robot trajectory must deviate from ζ^N. This results in a trajectory which is denoted by ζ_U. The deviation is in accordance with the position y and the shape and size of the obstacles; it is characterised by a Gaussian term in the UF with parameter θ. Ideally, a utility function estimated from demonstrations is desired to produce a similar behaviour in response to an added obstacle. Thus, we compute the parameters of the UF as per eq. (4) by minimising the distance between a trajectory generated by the UF in eq. (3) and the demonstrated trajectory. We align trajectories by dynamic time warping and build a signal of distances between corresponding points of the trajectories, i.e. $e(k) = \zeta_U(k) - \zeta_d(k)$ where y_d and θ_d are the position and parameters corresponding with the j_{th} obstacle.

$$\theta_i = \underset{\theta}{\operatorname{argmin}} \sum_{d=1}^{n_{dem}} \sum_{k}^{t_d} \left(\|e(k)\|_2 + \|e(k)\|_\infty\right) \tag{4}$$

In eq. (4), we assume that all the obstacles have identical shape and size, i.e. they belong to one object class. If there are more than one object class in the scene, θ for each obstacle class is computed by eq. (4). We also presume that the class of each obstacle is given, e.g. a computer vision algorithm provides a class label for each obstacle. For instance, two obstacles belonging to different object classes are used in the experiments with UR5 in section 6 whereas one obstacle is used in the experiments with Yumi.

We use a numerical optimisation approach (*minConf*) with a quasi-Newton strategy and limited-memory BFGS updates Schmidt (2010). Ghalamzan et al. (2015) proposed to use only L_2 norm of the error signal. This has a very small and even a zero gradient in some parts of the parameter space θ. Hence, the algorithm converges only with some initial conditions and the gradient-based optimisation method is not an appropriate approach for parameter computation. By definition, an obstacle has a local influence on the utility function and as a result, L_∞ norm of the error signal produces a non-zero gradient where ζ_U and ζ_d are different. If ζ_U converges to ζ_d, this L_∞ term goes to zero. In fact, it helps the convergence very much and has negligible influence on the computed optimal parameters. Consequently, we experienced a convergence behaviour much better than Ghalamzan et al. (2015) by adding L_∞ of the error signal in eq. (4). Even though demonstrations may be noisy, the recovered utility function is desired to generate noiseless trajectory.

5. Extension to dynamic environments

Similar to proposals in studies of sensorimotor learning in cognitive psychology and neuroscience (Todorov, 2004; Wolpert et al., 2011), we design the reproduction phase to cope with the moving obstacles. The approach proposed by Ghalamzan et al. (2015) is an open loop strategy in which the position of the obstacle is assumed to be known before generating a collision-free trajectory and it does not change during the whole experiment. By contrast, we postulate that a skill, which is learnt via a utility function for a static scene, can be generalised to a dynamic environment by extending the UF to include a feedforward and a feedback strategy. Thus, we separate a learning phase from the reproduction phase of our model, as shown in Figs. 5(a) and 5(b). This is very useful because the learning process occurs in a stationary scene, while the robot can perform the task in a dynamic environment by extending the model. Eventually, the robot can use this extended task model and perform the task in a very complex scene with many moving obstacles.

In Fig. 1, the robot is tasked with manipulating the white object while a human is moving a cylindrical red object (which we refer to as an obstacle). In Fig. 1(b), a human is moving the obstacle on the table. We use an off-the-shelf RGB camera to collect sensory information about the obstacle position during the experiment. This is used to build a real-time feedback control strategy. The camera is calibrated in the robot workspace using the camera calibration toolbox of MATLAB. An object tracker uses the images acquired by the camera and estimates the positions of the obstacle at each time, denoted by $\mathrm{y}_j(k)$. The object tracker applies a colour filtering algorithm, which is part of the OpenCV library. To filter the noisy signal provided by the object tracker, a Kalman filter (KF) is also designed to provide a noiseless estimation of the position of the object, denoted by $\hat{\mathrm{y}}_{j,0}(k)$, that eventually is used as a feedback control strategy in eq. (5).

$$\begin{aligned} U_{\mathrm{fb}}\left(\mathrm{x}_k : \zeta^N, \hat{\Omega}\right) &= \left(\mathrm{x}_k - \mathrm{x}_k^{\mathrm{N}}\right)^T \left(\mathrm{x}_k - \mathrm{x}_k^{\mathrm{N}}\right) + \sum_{j=1}^{n_{obs}} e^{\left(-\hat{\mathrm{f}}_{j,0}^T(k)\, \theta_j^{-1}\, \hat{\mathrm{f}}_{j,0}(k)\right)} \\ \hat{\mathrm{f}}_{j,0}(k) &= \mathrm{x}_k - \hat{y}_{j,0}(k) \end{aligned} \tag{5}$$

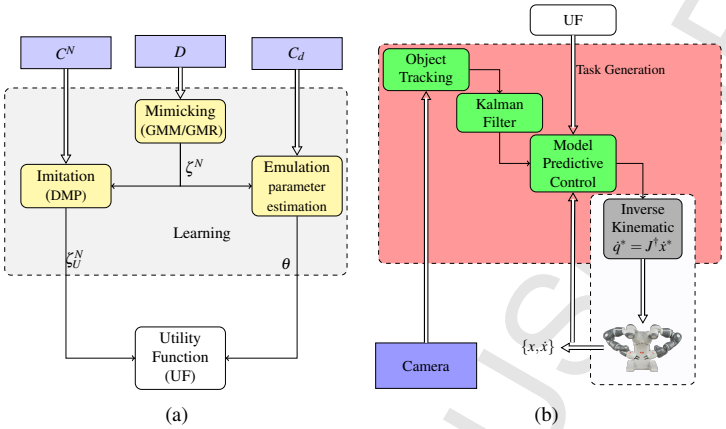


Figure 5: A scheme of the incremental robot learning from demonstration inspired by observational learning. In Fig. (a), $D = \{\zeta_1, ..., \zeta_{N_{dem}}\}$ is a set of demonstrated trajectories of a task. There are three levels of learning from observation/demonstrations: mimicking: a learner builds a model which is the mean of the observations, denoted by ζ^N. Imitation: the mean of the observations is then generalised, denoted by ζ_U^N, to satisfy C^N which is different from the demonstrations. Emulation: the trajectoryt generated via imitation is used in combination with the environmental features to build a Utility Function (UF). The UF can generate a trajectory that satisfies C_d, denoted by ζ_U. Fig. (b) shows the generation schematic that extends the approach to deal with a moving obstacle. This includes a camera, an object tracker, a Kalman filter, a model predictive and an inverse kinematic block. More details of these blocks are presented in Fig. 6(b). YuMi image is taken from http://cobotsguide.com/2016/06/abb-yumi/.

where $\hat{\Omega} = \{\hat{O}_1, ..., \hat{O}_{n_{obs}}\}$ and $\hat{O}_j = \{\hat{y}_{j,0}(k), \theta_j\}$. A prediction of the future positions of the object is then included in the UF which we refer to as feedforward. The predictions are provided by a linear model of the KF, denoted by $\{\hat{y}_{j,1}(k), ..., \hat{y}_{j,t_P}(k)\} = KF(y_j(k))$ where t_P is the prediction time horizon. However, the obstacle movements may not be linear. Thus, an uncertainty about the obstacle position is included in the UF by incorporating a summation of the UF over some samples of the predicted positions; the samples are normally distributed around the values computed by the KF and denoted by $\tilde{y}_{j,t,s}(k) = \mathcal{N}(\hat{y}_{j,t}(k), \sigma_t)$ where $t = 1, ..., t_P$. The uncertainty bound is characterised by $\sigma_t = \sigma_a \times t + \sigma_b$ where σ_a and σ_b are the parameters of the uncertainty and can be set empirically. In Fig. 6(a), a sample utility function with uncertainty bound (solid lines) and without uncertainty bound (dashed lines) are shown. In this figure, the prediction time horizon is 5. The longer the prediction time, the bigger the uncertainty bound. This implies that we have less confidence in the predicted values with a larger prediction time. Finally, we use these samples in the UF as per eq. (6) to form the feedforward control strategy based on the model learnt from demonstrations.

$$U_{\text{ff}}\left(x_k : \zeta^N, \tilde{\Omega}\right) = \sum_{j=1}^{n_{obs}} \sum_{t=1}^{t_p} \frac{1}{n_s} \sum_{s=1}^{n_s} e^{\left(-\tilde{f}_{j,t,s}^T(k)\, \theta_j^{-1}\, \tilde{f}_{j,t,s}(k)\right)} \tag{6}$$

where $\tilde{\Omega} = \{\tilde{O}_1, ..., \tilde{O}_{n_{obs}}\}$ and $\tilde{O}_j = \{\tilde{y}_{j,t,s}(k), \theta_j\}$; $\tilde{f}_{j,t,s}(k) = x_k - \tilde{y}_{j,t,s}(k)$ and $\tilde{y}_{j,t,s}(k)$ is the position of the s_{th} sample of the j_{th} obstacle at t_{th} step of the prediction horizon at the time step k. The total UF combines the feedback and feedforward UF by summing

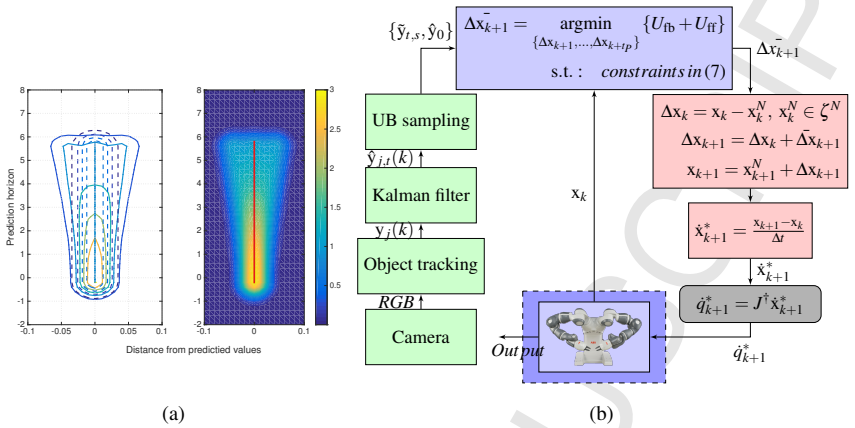


Figure 6: The left figure in (a) shows the level set of the UF with (line) and without (dashed line) uncertainty bound. The right figure in (a) shows the UF at time k along a prediction horizon with the considered uncertainty bound that increases with the prediction time horizon. Fig. (b) shows a scheme of the task generation block combining feedback and feedforward control strategy. It is used to compute an optimal solution at each time k in the presence of a moving obstacle. YuMi image is taken from http://cobotsguide.com/2016/06/abb-yumi/

eq. (5) and (6). Finally, we find the solution to the combined UF by a Model Predictive Control (MPC) strategy with some constraints as per eq. (7).

$$\bar{\mathrm{x}}(k) = \underset{\{\mathrm{x}(k),\mathrm{x}(k+1),\ldots,\mathrm{x}(k+t_P)\}}{\operatorname{argmin}} \left\{ \overbrace{U_{\mathrm{fb}}\left(\mathrm{x}(k) : \zeta^N, \hat{\Omega}\right)}^{Feedback} + \overbrace{U_{\mathrm{ff}}\left(\mathrm{x}(k+t) : \zeta^N, \tilde{\Omega}\right)}^{Feedforward} \right\}$$

$$\begin{aligned}
&\text{s.t. :} \\
&\forall\, t = 0,1,\ldots,t_p;\ s = 1,\ldots,n_s;\ j = 1,\ldots,n_{obs} \\
&\mathrm{x}(k);\ \mathrm{O}_j(k), && Measurement, \\
&\{\hat{\mathrm{y}}_{j,0}(k),\ \hat{\mathrm{y}}_{j,1}(k),\ldots,\hat{\mathrm{y}}_{j,t_P}(k)\} = KF(\mathrm{y}_j(k)) && Prediction\ by\ KF, \\
&\tilde{\mathrm{y}}_{j,t,s}(k) = \mathcal{N}(\hat{\mathrm{y}}_{j,t}(k), \sigma_t) && Samples\ in\ UB, \\
&\sigma_t = \sigma_a \times t + \sigma_b && UB\ model, \\
&\mathrm{x}(k+t) = \mathrm{x}(k-1+t) + \mathrm{a}(k+t) && State\ transition, \\
&\mathrm{x}(k+t) \in \mathscr{X} && State\ constraints, \\
&\mathrm{a}(k+t) \in \mathscr{U} && Action\ constraints.
\end{aligned} \tag{7}$$

where $k = 1,\ldots,H$ and H is the time to complete the task according to the demonstration. This formulation is well suited for episodic manipulation tasks with a continuous state space and discrete time. At each time step k, the space to be searched for the optimal solution is the cotangent space of ζ^N_{dmp} at x^N_k as shown in Fig. 4(a), i.e. $\mathrm{x}(k) \in \mathrm{N}_k$.

6. Experimental Result

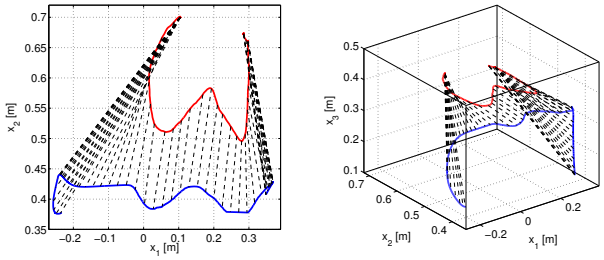
To show that the model, which is learnt from the demonstration for performing the task and avoiding obstacles, is highly important for performing many robotic tasks,

we present two experiments whose datasets collected with a YuMi robot and a UR5 robot. The first experiment is pick-and-place with YuMi (Fig. 1(a)) and the second one is sweeping with UR5 (Fig. 10(a), 10(b) and 10(c)). The dataset in Fig. 1(c) and 10(d) are collected by *kinesthetic* teaching. In kinesthetic teaching, a human user pushes and pulls a robot arm and moves it to a few waypoints. Then, the robot uses these demonstrated waypoints to generate a smooth trajectory while it is collecting the corresponding data.

We assume these set of demonstrated trajectories are given, denoted by $\zeta_d \subset \mathscr{D}, \forall\, d = 1,...,n_{dem}$. In addition, we presume there are utility functions for every environment that can generate trajectories that are very close to the demonstrations. Therefore, the optimal solution to the utility function satisfies both tasks and scene constraints. The corresponding utility function is estimated and used to generate the trajectories $\zeta_{U,d}$ close to those demonstrated (ζ_d) that satisfy the corresponding task and scene constraints.

Pick-and-place experiment with YuMi aims at demonstrating the contribution of this paper according to the motivating example presented in the introduction (section 1); whereas the sweeping experiment with the UR5 robot is presented for the sake of completeness of the description. In the Yumi experiment, the robot must pick up the white object, shown in Fig. 1(b), from an initial location, which is fixed on a table, and then place it in the blue box on the table while avoiding colliding with the red object. We demonstrate the task (Fig. 1(a)) once without any obstacles and once with the presence of the red object. In this example, the robot is supposed to perform the task where (1) the position of the box is changed and while (2) a cylindrical red object is being moved in the robot's workspace. This experiment with YuMi illustrates the effectiveness of IRLfD-DE on performing a task in an environment with a moving obstacle. Of note, the noisy data set collected by the UR5 robot does not include the nominal trajectories (Fig. 10(d)); whereas the nominal trajectory is present in the Yumi dataset (Fig. 1(c)). The demonstrated trajectories with and without the stationary obstacle, namely ζ_d and ζ_N, are shown in Fig. 1(c) with black and red lines, consecutively. The UF is built based on the demonstrations where the reproduction error and the accuracy achieved using the computed model are[2] $MSE_R = 5.7168 \times 10^{-5}\ [m]$ and $Pr = 76.96$. The number of sample points of the demonstrated trajectory is $H = 2596$. Fig. 7 shows the pick-and-place experiment with the Yumi ABB robot. Fig. 7(a) and 7(b) show the correspondence between the position of the red and white objects. This illustrates how our approach effectively enables the robot to avoid colliding between white object and the moving obstacle (red object). We performed the experiment five times in total while the human was intentionally trying to move the object in the robot's workspace and cause a collision between the white and red objects. These different experiments aim at illustrating the capability of the IRLfD-DE to cope with such moving obstacles in real human-robot shared workspaces with different velocity profiles of the obstacle. As shown in Fig. 8, the robot successfully generalises the task at two levels: (i) at imitation, it puts the object in the box at a new position (in Fig. 8, the terminal

[2] $Pr = \frac{MSE_d - MSE_U}{MSE_d} \times 100$ where $MSE_d = \frac{1}{t_d}\sum_{k=1}^{t_d} \|\zeta_d(k) - \zeta^N(k)\|^2$ and $MSE_U = \frac{1}{t_d}\sum_{k=1}^{t_d} \|\zeta_d(k) - \zeta_U(k)\|^2$. For more details, please see (Ghalamzan et al., 2015).



(a) (b)

Figure 7: (a) and (b) show the top and perspective view of the trajectories of the robot's end-effector (blue line) during task execution and the moving obstacle (red line). These figures show the robot successfully avoids colliding with the obstacle which is moved by a person on the table to intentionally cause collision. The black dashed lines show the correspondence between the position of the robot's end-effector and the obstacle at every time step.

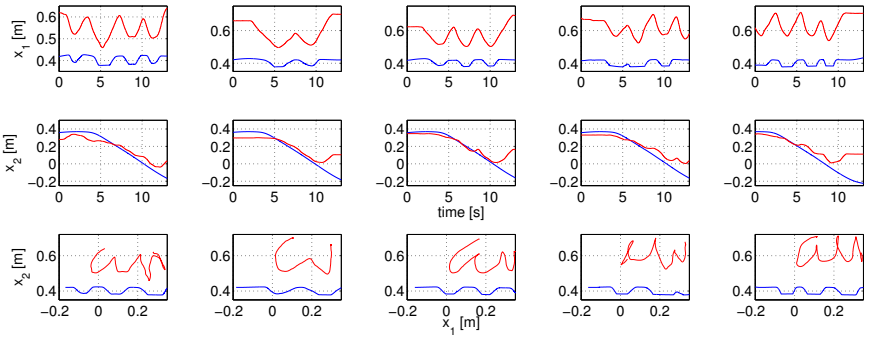


Figure 8: The end effector trajectories of the pick-and-place task performed with YuMi robot (blue line) and the corresponding trajectories of the red object. The first forth rows at left show the robot successfully complete the task while avoiding the moving red cylinder. The very right row of figures show the robot put the object in the box at a new position while avoiding the moving obstacle. In these experiments, it is tried to demonstrate different collision behaviours by moving the red object with different velocities showing the performance of IRLfD-DE in coping with such situations. At some parts the x_1 cannot be changed further because the robot has reached the joint limits, e.g. in the first figure at right at $t = 5 - 7[s]$, x_1 cannot be further decreased.

point of the trajectory in the right row is different than the trajectories presented in other rows of the figure) and (ii) at emulation, it avoids colliding with the moving red cylindrical object in the robot workspace (Fig. 8). Fig. 9(a) shows the velocities of the red object. In addition, Fig. 9(b) is the box plot of the velocity of the red object across different experiments. The white object is manipulated along x_2 while the red object is approaching it mostly along x_1. Hence, the approach velocity is mostly represented by the velocity of the red object. This figure shows that our proposed approach successfully avoids collision with the moving obstacle whose velocities are in the range

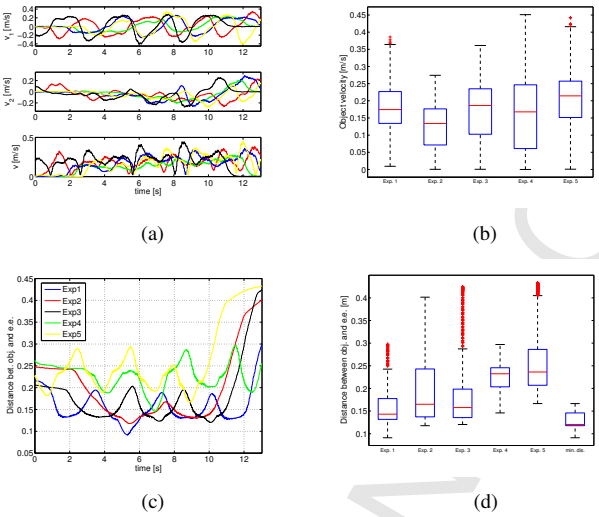


Figure 9: (a) shows the velocities of the red object during performing the experiment; (b) is the box plot of the velocity of the red object across different experiments. The white object is manipulated along x_2 while the red object is approaching it mostly along x_1. Hence, the approach velocity is mostly represented by the velocity of the red object. This figure shows that our approach successfully avoids collision with the moving obstacle with velocities in the range of 0.1 to 0.45 $[m/s]$; 9(c) shows trajectories of the corresponding distance of the robot's end-effector and the red object. By comparing the corresponding distances and the velocities of the red object one can see the minimum distance in some cases are less than the demonstrated distance. For instance, at time $5-6[s]$ the distance of the first experiment is less than the demonstrated one ($0.11cm$) because the robot reached the joint limit and cannot move further in the direction needed for avoiding the obstacle; (d) shows the box plot of the distances in (c). The distribution of the minimum distances shows that our approach successfully could avoid collision with the moving obstacle by keeping the distance between the end effector and the red object according to the demonstration.

of 0.1 to 0.45 $[m/s]$; the effectiveness of our approach may be better understood by Fig. 9(c) showing the corresponding distance of the robot's end-effector and the red object. By comparing the corresponding distances and the velocities of the red object one can see the minimum distance in some cases are less than the demonstrated distance. For instance, at time $5-6[s]$ the distance of the first experiment is less than the demonstrated one ($0.11cm$) because the robot reached the joint limit and cannot move further in the direction needed for avoiding the obstacle. Fig. 9(d) shows the box plot of the distances represented in Fig. 9(c). The distribution of the minimum distances, which is in the range of $[0.09, 0.15]$ cm, shows that our approach could successfully cope with the very nonlinear movements of the obstacle. These nonlinear movements shown in Fig. 9(c) characterise the very complex dynamic of the object movements.

In terms of practical aspects of the implementation, we considered two separate processes here: (1) hard real-time component, (2) soft real-time component. For the computation process, we use a PC with a quad-core Intel© Core® i3-2120 processor, 8 GB of RAM and a real-time Linux-based operating system. The soft real-time component is labelled by the "Task Generation Phase" (TGP) in Fig. 5. In this component,

exec. time	
mean [ms]	1.5034
std [ms]	1.3327
min [ms]	0.6500
max [ms]	16.7030

Table 1: TGP process execution time

an object tracker detects a moving obstacle and estimates its position, which is then filtered by a Kalman filter (KF). The KF also provides a set of predicted positions of the object at every time step k. This information is then used in the MPC-based optimisation problem in eq. (7). The optimisation is solved by COBYLA, a nonlinear optimisation algorithm (Powell, 1994) that can compute the solution very quickly. We empirically checked the computation speed which illustrates that the computation of the solution to the eq. (7) by the COBYLA algorithm satisfies the necessary soft real-time constraint. In particular, table 1 reports the statical information about the measured execution times of a population of $10,000$ complete iterations of the TGP, i.e. object detection, tracking, Kalman filter and nonlinear optimisation. Table 1 shows that the average of the execution time is 1.5034 [ms] with a standard deviation of 1.3327 [ms]. According to the study presented in Miller response to control activation, e.g. clicking of the typewriter key, which is no more than 100 [ms] is considered to be instantaneous. Furthermore, the highest frequency at which a human can perceive visual information is 13 [ms]. Here, we consider a time-frequency of 20 [ms] to be enough for real-time collaboration between a human and a robot. The maximum and minimum time durations required for the TGP computation are 16.7030 [ms] and 0.6500 [ms]. Therefore, we set the time for computing the TGP to be 20 [ms], i.e. 50 Hz, which is necessary for a human to perceive it as a real-time process. On the other hand, the second process is hard real-time, which is shown with the grey block labelled as "Inverse Kinematics" in Fig. 5(b) and 6(b). This process communicates with the robot at a frequency of 250 Hz, necessary for the robot's real-time controller. This process sends the robot the necessary joint positions required for the low-level joint control of the robot. Although the IK works at 250 Hz, the TGP provides the IK with the updated joint positions at the frequency of 20 Hz. In order to bridge between the values at different frequencies, a first-order hold is utilised at 250 Hz that interpolates the consecutive Cartesian points computed by the TGP. This makes IK generate a smooth trajectory. The IK will send a piecewise constant Cartesian trajectory to the robot if the first-order hold is not used.

Here, we also present one example of *sweeping experiment* (which was fully presented in (Ghalamzan et al., 2015)) to show that using IRLfD-DE in a scene with multiple obstacles is straightforward. In this experiment, the UR5 industrial manipulator learns how to sweep an object into a dustpan from a few demonstrations while avoiding colliding with two types of objects. All the objects are located on the table shown in Fig. 10. The object to be swept is in front of the dustpan and the obstacles, namely the cup and the marker (Fig. 10(a) and 10(b)), are in the way of the robot, during demonstrations. *The Sweeping task* is demonstrated twice while avoiding a collision with the marker (Fig. 10(a)) and twice while avoiding colliding with the cup (Fig. 10(b)). The

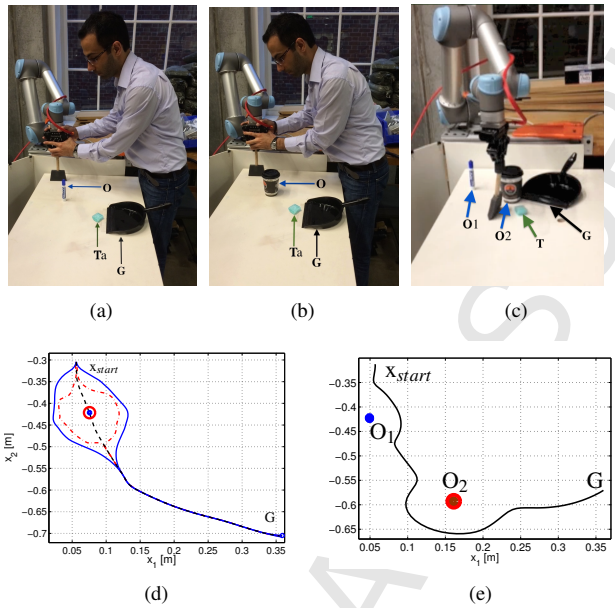


Figure 10: A human is demonstrating how to sweep the green cube (Ta) into the dustpan (G) while avoiding colliding with obstacles, namely a marker shown in (a) and a cup shown in (b), by moving the UR5 robot arm. This example illustrates that the robot learns a model of the task and obstacle avoidance policy from demonstration and then performs the sweeping task by successfully combining different obstacle avoidance policy with the task model. Fig. (c) shows a scene of sweeping example with a new location of the dustpan (shown with G), the marker and the cup (shown with O_1 and O_2). Both task and scene constraints are different from those demonstrated; (d) shows the demonstrated trajectories for the sweeping task. The red dashed and blue lines correspond with Fig. 10(a) and 10(b), respectively. 10(e) shows the trajectories generated using the corresponding UF corresponding with the scene shown in (c). The trajectory successfully satisfies the task constraint and avoids colliding with both obstacles in the scene. (Ghalamzan et al., 2015)

corresponding trajectories are shown with the red dashed and blue line in Fig. 10(d). The Gaussian process is then used (according to the first level of IRLfD) to compute an average path ζ^N from the demonstrations. A DMP model trained by this trajectory generates the necessary trajectory (ζ^N_{dmp}) for sweeping the object into a dustpan at a new position shown with G in Fig. 10. The parameter of a utility function for avoiding the obstacles, namely the marker and the cup, is learnt from demonstrated trajectories. Finally, the UF is built according to ζ^N_{dmp} and positions of the obstacle in the scene as in section 5. The utility function yields a new trajectory that satisfies both the task and scene constraints. The constraints imposed by the objects in the scene, namely the marker and coffee cup, are shown consecutively with O_1 and O_2 in Fig. 10. According to the demonstrations, we expect the UF generates a different response to O_1 and O_2, as shown in Fig. 10(e). The cup results in a deviation from ζ^N_{dmp} which is larger than the one caused by the marker. This example illustrates that different obstacle avoidance policy learnt separately from different demonstrations are combined in a single UF. This allows keeping the learning process very simple; despite the task execution

which can be complex. This experiment is fully discussed in (Ghalamzan et al., 2015); nonetheless, we briefly presented it here to show the performance of IRLfD in the presence of two different types of obstacles. Because performing the pick-and-place task with multiple moving obstacles is not feasible due to the limited workspace of the YuMi robot, we presented the sweeping task with two different obstacles to show the feasibility of using the IRLFD-DE to cope with the multiple moving obstacles.

7. Discussion

In our approach, namely IRLfD-DE, the robot learns a model of a task from demonstrations in a stationary scene and IRLfD-DE generates trajectories necessary to perform the task at different levels of generalisation: (1) it computes a noiseless trajectory from noisy observations, denoted by ζ^N, at the mimicking level; (2) it computes ζ^N_{dmp} that satisfies task constraint at the imitation level; (3) it computes ζ_U that satisfies the constraints imposed by the task and the scene, denoted by C_n and C_d, according to the imitation and emulation level of IRLfD; (4) it computes a trajectory that satisfies the varying constraints of the task and the scene, i.e. a scene with a moving obstacle.

This paper is accompanied by a video of two experiments: (1) sweeping with the UR5 robot and (2) manipulation with the ABB YuMi robot. The video shows the complete processes of RLfD. First, the task is demonstrated with a stationary obstacle at every demonstration. Then, the robot learns a model of task and obstacle avoidance from demonstrations and replicates the sweeping task in the presence of two stationary obstacles. The pick-and-place experiment with YuMi is then presented, including (i) a human demonstrates the task with and without obstacle and (ii) the robot performs the task while a human is moving an obstacle in the robot's workspace. These experiments show how the proposed approach facilitates human-robot collaboration such that a user can easily teach the desired task as well as desired obstacle avoidance control policies to an out-of-the-cage robot. The experiments also prove the effectiveness of our approach in terms of learning a model from simple task demonstrations in the presence of only a single stationary obstacle each time where the robot generalises this model and perform the task in complex situations with moving obstacles. Of note, it may not even be possible to demonstrate the task with a moving obstacle using kinesthetic teaching. Although we did not experiment with more than one moving obstacle, utilising the approach for such an example is straightforward. The computation complexity for adding other moving obstacles increase linearly with the number of obstacles, i.e. it is $n_{obs}C(IRLfD-DE)$ where n_{obs} is the number of obstacles and $C(IRLfD-DE)$ is the computation complexity for using IRLfD-DE model for performing the task while one obstacle is moving in the robot's workspace.

The results of the examples presented in this paper support the premise stated in the introduction. This is practically useful for a common situation in the human-robot collaboration context. An increasing interest in HRC and the desire to use robots out-of-the-cage in collaboration with human workers are the main motivation for this work. A human co-working with a robot may need to change a robot's behaviour from time to time. Here, by the behaviour we mean the task and obstacle avoidance control policy. We also discussed the computation costs of the reproduction phase, illustrating the feasibility of the proposed approach in real time.

Although this approach can already be used in many manipulation tasks, extending it to tackle a higher dimensional problem will be an interesting future work. In addition, it is appealing to study this approach jointly with configuration space obstacle avoidance, where the latter does not need to be learnt from demonstration and can be programmed in advance as proposed by Ratliff et al. (2009). What is the maximum distance that the robot's end-effector can deviate from a given nominal trajectory in a dynamic environment? The answer to this question creates a further intriguing future study.

For instance, the YuMi robot reaches the workspace limits at some segments of the trajectory where $x_1 = 0.48$ [m] in Fig. 7(a) because the human intentionally tries to cause as much deviation as possible. One example strategy may be to stop performing the task if there is such an interfering element in the scene. For example, if the total optimal UF value of the next step exceeds a predefined threshold, the robot stops. Lastly, a study on using a probabilistic approach for the emulation utility function is also of interest for future research; therefore, any deviation from the nominal environment can be considered in the learning emulation UF. Moreover, one can investigate the online adaptation of the generated trajectory to a new target point in the proposed approach, as it has been studied by Pastor et al. (2009). Lastly, this approach can also be investigated for grasping and manipulation problems in combination with methods (e.g. Ghalamzan et al., 2016) developed for selecting the optimal grasp.

8. Conclusion

In this paper, we extended the approach to robot learning from demonstrations presented by Ghalamzan et al. (2015) such that it can be used to generate the necessary trajectory to satisfy the constraints of avoiding a collision with a moving obstacle. Hence, a model of the task trajectory and the response to a stationary obstacle is learnt from demonstrations, as a robot can use the model to perform the task in a dynamic environment. By dynamic environment, here, we refer to an environment in which the constraints of the task and the environment can change. For instance, in the pick-and-place example, the target point of the trajectory changes at each task execution; whereas the obstacle moves during task execution. Although one may use a unified framework (e.g. Levine et al., 2016; Gu et al., 2016), providing an end-to-end RLfD solution to the problem presented in this paper, such an approach would become hard to interpret and more difficult to use for a different task. Here, we propose incremental robot learning from the demonstration for dynamic environment (IRLfD-DE) that is easy to use, interpret and evaluate. The modules of IRLfD-DE provide us with different levels of generalisation which correspond with observational learning in human studies. We demonstrate the effectiveness and usefulness of the approach with a practical example of pick-and-place by a YuMi ABB robot. This experiment illustrates how a user can easily determine the behaviour of the robot by using this approach. In addition, we show how the approach can handle multiple stationary obstacles in an experiment.

Abbeel, P., Ng, A. Y., 2004. Apprenticeship learning via inverse reinforcement learn-

ing. In: Proceedings of the international conference on Machine learning. ACM, pp. 1 – 9.

Argall, B., Chernova, S., Veloso, M., Browning, B., 2009. A survey of robot learning from demonstration. Robotics and autonomous systems 57 (5), 469–483.

Billard, A., Calinon, S., Dillmann, R., 2016. Learning from humans. In: Springer Handbook of Robotics. Springer, pp. 1995–2014.

Billard, A., Calinon, S., Dillmann, R., Schaal, S., 2008. Survey: Robot programming by demonstration. Tech. rep., MIT Press.

Byravan, A., Monfort, M., Ziebart, B., Boots, B., Fox, D., 2015. Graph-based inverse optimal control for robot manipulation. In: IJCAI International Joint Conference on Artificial Intelligence. Vol. 2015-January. pp. 1874–1880.

Calinon, S., 2009. Robot Programming by Demonstration. EPFL Press.

Calinon, S., Sardellitti, I., Caldwell, D. G., 2010. Learning-based control strategy for safe human-robot interaction exploiting task and robot redundancies. In: Proceedings of International Conference on Intelligent Robots and Systems. IEEE, pp. 249–254.

Fajen, B. R., Warren, W. H., 2003. Behavioral dynamics of steering, obstable avoidance, and route selection. Journal of Experimental Psychology: Human Perception and Performance 29 (2), 343.

Flanagan, J. R., Vetter, P., Johansson, R. S., Wolpert, D. M., 2003. Prediction precedes control in motor learning. Current Biology 13 (2), 146–150.

Gams, A., Denisa, M., Ude, A., 2015. Learning of parametric coupling terms for robot-environment interaction. In: Proceedings of International Conference on Humanoid Robots. IEEE, pp. 304–309.

Ghalamzan, A. M., Mavrakis, N., Kopicki, M., Stolkin, R., Leonardis, A., 2016. Task-relevant grasp selection: A joint solution to planning grasps and manipulative motion trajectories. In: Proceedings of IEEE/RS International Conference on Intelligent Robots and Systems. IEEE, pp. 907–914.

Ghalamzan, A. M., Paxton, C., Hager, G. D., Bascetta, L., 2015. An incremental approach to learning generalizable robot tasks from human demonstration. In: Proceedings of International Conference on Robotics and Automation. IEEE, pp. 5616–5621.

Gu, S., Holly, E., Lillicrap, T., Levine, S., 2016. Deep reinforcement learning for robotic manipulation. arXiv preprint arXiv:1610.00633.

Guenter, F., Hersch, M., Calinon, S., Billard, A., 2007. Reinforcement learning for imitating constrained reaching movements. Advanced Robotics 21 (13), 1521–1544.

Hoffmann, H., Pastor, P., Park, D., Schaal, S., 2009. Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance. In: Proceedings of International Conference on Robotics and Automation. IEEE, pp. 2587–2592.

Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., Schaal, S., 2011. Stomp: Stochastic trajectory optimization for motion planning. In: Proceedings of International Conference on Robotics and Automation. IEEE, pp. 4569–4574.

Karaman, S., Frazzoli, E., 2010. Incremental sampling-based algorithms for optimal motion planning. Robotics Science and Systems VI 104.

Kober, J., Wilhelm, A., Oztop, E., Peters, J., 2012. Reinforcement learning to adjust parametrized motor primitives to new situations. Autonomous Robots 33 (4), 361–379.

Kormushev, P., Calinon, S., Caldwell, D. G., 2010. Robot motor skill coordination with em-based reinforcement learning. In: Proceedings of International Conference on Intelligent Robots and Systems. IEEE, pp. 3232–3237.

Krug, R., Dimitrov, D., 2015. Model predictive motion control based on generalized dynamical movement primitives. Journal of Intelligent & Robotic Systems 77 (1), 17–35.

Levine, S., Finn, C., Darrell, T., Abbeel, P., 2016. End-to-end training of deep visuomotor policies. Journal of Machine Learning Research 17 (39), 1–40.

Maeda, G., Ewerton, M., Neumann, G., Lioutikov, R., Peters, J., 2017. Phase estimation for fast action recognition and trajectory generation in human–robot collaboration. The International Journal of Robotics Research, –.

Miller, R. B., 1968. Response time in man-computer conversational transactions. In: Proceedings of the December 9-11, 1968, fall joint computer conference, part I. ACM, pp. 267–277.

Ng, A. Y., Russell, S. J., 2000. Algorithms for inverse reinforcement learning. In: Proceedings of the international conference on Machine learning. pp. 663–670.

Park, D., Hoffmann, H., Pastor, P., Schaal, S., 2008. Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In: Proceedings of International Conference on Humanoid Robots. IEEE, pp. 91–98.

Pastor, P., Hoffmann, H., Asfour, T., Schaal, S., 2009. Learning and generalization of motor skills by learning from demonstration. In: Proceedings of International Conference on Robotics and Automation. IEEE, pp. 763–768.

Powell, M. J. D., 1994. A direct search optimization method that models the objective and constraint functions by linear interpolation. Advances in Optimization and Numerical Analysis, 51–67.

Rai, A., Meier, F., Ijspeert, A., Schaal, S., 2014. Learning coupling terms for obstacle avoidance. In: Proceedings of International Conference on Humanoid Robots. IEEE, pp. 512–518.

Rai, A., Sutanto, G., Meier, F., Schaal, S., 2016. Learning feedback terms for reactive planning and control. arXiv preprint arXiv:1610.03557.

Ratliff, N., Zucker, M., Bagnell, J. A., Srinivasa, S., 2009. Chomp: Gradient optimization techniques for efficient motion planning. In: Proceedings of International Conference on Robotics and Automation. IEEE, pp. 489–494.

Schaal, S., 1999. Is imitation learning the route to humanoid robots? Trends in cognitive sciences 3 (6), 233–242.

Schmidt, M., 2010. Graphical model structure learning with l1-regularization. Ph.D. thesis, University of British Columbia.

Tanner, W. R., 1981. Industrial Robots: Fundamentals. Society of Manufacturing Engineers.

Tanwani, A. K., Calinon, S., 2016. Learning robot manipulation tasks with task-parameterized semitied hidden semi-markov model. IEEE Robotics and Automation Letters 1 (1), 235–242.

Thompson, D. E., Russell, J., 2004. The ghost condition: imitation versus emulation in young children's observational learning. Developmental Psychology 40 (5), 882 – 899.

Thoroughman, K. A., Shadmehr, R., 2000. Learning of action through adaptive combination of motor primitives. Nature 407 (6805), 742–747.

Todorov, E., 2004. Optimality principles in sensorimotor control. Nature neuroscience 7 (9), 907–915.

Wheeler, D. S., Fagg, A. H., Grupen, R., 2002. Learning prospective pick and place behaviour. In: Proceedings of the 2nd International Conference on Development and Learning. IEEE, pp. 197–202.

Whiten, A., Ham, R., 1992. On the nature and evolution of imitation in the animal kingdom: reappraisal of a century of research. Advances in the Study of Behaviour 21 (1), 239–283.

Whiten, A., McGuigan, N., Marshall-Pescini, S., Hopper, L. M., 2009. Emulation, imitation, over-imitation and the scope of culture for child and chimpanzee. Philosophical Transactions of the Royal Society B: Biological Sciences 364 (1528), 2417 – 2428.

Wolpert, D. M., Diedrichsen, J., Flanagan, J. R., 2011. Principles of sensorimotor learning. Nature Reviews Neuroscience 12 (12), 739–751.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., Dey, A. K., 2008. Maximum entropy inverse reinforcement learning. In: AAAI. pp. 1433–1438.

Amir-Masoud Ghalamzan-Esfahani was born in Tehran-Iran in 1981. He received his M.Sc. in Mechanical Engineering from Iran University of Science and Technology in 2009. He then completed his study in Second Level Specialisation in Automatic Control Engineering 'Cum Lude' at Politecnico di Torino in 2011. From 2013 to 2015 he has been a PhD student in the department of Information Technology at Politecnico di Milano. He defended his thesis titled "Towards Robot learning from demonstration". He joined the University of Birmingham as a Postdoc fellow in August 2015 where he is working on robotic manipulation. His research interest includes robotic manipulation, automatic control, robot learning from demonstration, path planning, optimisation and human-robot collaboration.

Matteo Ragaglia was born in Piacenza (Italy) in 1988.

He received the Bachelor of Science "cum laude" in Computer Science Engineering at Politecnico di Milano in September 2010, defending a thesis on the use of Android-based mobile devices inside a dometic environment in order to help disabled people.

In December 2012 he obtained his Master of Science in Computer Science Engineering at Politecnico di Milano defending a thesis regarding the online computation of severity indices for safe human-robot cooperation. From January 2013 to December 2015 he attended the PhD program in Information Technology at Politecnico di Milano. He earned his PhD Degree "with merit" in February 2016, defending a thesis entitled "Towards a safe interaction between humans and industrial robots through perception algorithms and control strategies". In December 2015 he joined Yanmar R&D Europe, where he is currently developing research activities in the field of robotics, construction machines and human-machine interfaces. His research interests include automatic control, industrial robotics, safe human-robot cooperation and path planning strategies for autonomous vehicles.

Robot Learning from Demonstrations: Emulation Learning in Environments with Moving Obstacles?

This paper extends the approach to robot learning from demonstrations presented by Ghalamzan et al. (2015) such that it can be used to generate the necessary trajectory to satisfy the constraints of avoiding a collision with a moving obstacle. Hence, a model of the task trajectory and the response to a stationary obstacle is learnt from demonstrations, as a robot can use the model to perform the task in a dynamic environment. By dynamic environment, here, we refer to an environment in which the constraints of the task and the environment can change. For instance, in the pick-and-place example, the target point of the trajectory changes at each task execution; whereas the obstacle moves during task execution. Although one may use a unified framework (e.g. Levine et al., 2016; Gu et al., 2016), providing an end-to-end RLfD solution to the problem presented in this paper, such an approach would become hard to interpret and more difficult to use for a different task. Here, we propose incremental robot learning from the demonstration for dynamic environment (IRLfD-DE) that is easy to use, interpret and evaluate. The modules of IRLfD-DE provide us with different levels of generalisation which correspond with observational learning in human studies. We demonstrate the effectiveness and usefulness of the approach with a practical example of pick-and-place by a YuMi ABB robot. This experiment illustrates how a user can easily determine the behaviour of the robot by using this approach. In addition, we show how the approach can handle multiple stationary obstacles in an experiment.